VGN: VALUE-GUIDED NAVIGATION FROM RGB FOR ROBOT MOTION PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual navigation from monocular RGB observations remains challenging due to the need for reliable scene understanding and safe decision-making in diverse real-world environments. Existing modular approaches often depend on additional sensing or interaction-heavy pipelines for reliable traversability estimation, while end-to-end methods typically infer navigation-relevant structure indirectly through trajectory-centric supervision. Motivated by these observations, we propose Value-Guided Navigation (VGN), an RGB-based framework for learning dense traversability-aware and goal-conditioned planning representations for downstream navigation. To achieve this, we leverage vision foundation models together with classical planning algorithms to generate dense traversability and cost-to-go pseudo-labels directly from static RGB images. This supervision is generated without trajectory supervision or robot interaction and is used to train the student planning representation model. The learned representation is planner-agnostic and can be integrated with multiple downstream planning mechanisms. Experiments across multiple outdoor datasets, closed-loop simulation environments, and real-world deployment on a Unitree Go2 robot demonstrate effective RGB-based traversability estimation and local motion planning.

1 INTRODUCTION

Recently, visual navigation has emerged as a fundamental problem in robotics, attracting growing interest from both the robotics and computer vision communities (Shah et al., 2023b; Wortsman et al., 2019). By relying primarily on visual observations for environment understanding, visual navigation can reduce dependence on additional sensing for perception and local planning. However, robust operation in diverse real-world environments still requires reliable scene understanding and safe decision-making. Existing visual navigation systems generally follow either modular or end-to-end paradigms (Pushp et al., 2025). Modular approaches typically adopt a sense-plan-act pipeline. These systems first estimate the environment's traversability or semantic representations, and subsequently generate trajectories using classical optimization-based or sampling-based planners (Wellhausen & Hutter, 2021; Cao et al., 2022). Such approaches are often interpretable and can generalize well across environments. However, learning reliable traversability representations often depends on large-scale robot interaction, temporal aggregation, or additional sensing and state estimation pipelines in diverse real-world environments (Gasparino et al., 2022; Frey et al., 2024; Jung et al., 2024). Moreover, explicit map construction and planning stages may introduce additional computational overhead and latency (Roth et al.). In contrast to modular pipelines, recent learning-based navigation methods directly predict navigation actions, waypoints, or trajectories from visual observations, enabling adaptive and reactive navigation behaviors (Ramesh et al., 2026; Cai et al.). Existing approaches are commonly trained through imitation learning (IL), reinforcement learning (RL), or optimization-based learning paradigms such as imperative learning (ImpL) (Shah et al., 2023a; Kulkarni & Alexis, 2024; Yang et al., 2023). IL-based methods learn navigation behaviors from expert demonstrations, requiring navigation-relevant scene understanding to be inferred indirectly from demonstrated navigation behavior, which can limit generalization when demonstration diversity is limited (Sridhar et al., 2024; Inglin et al., 2026). RL-based approaches alleviate the need for expert demonstrations, but often require large-scale simulation environments, carefully designed reward functions, and extensive interaction data for stable training (Truong et al., 2023; Wang et al., 2026). More recent optimization-driven approaches based on ImpL improve trajectory prediction through differentiable planning objectives and geometric cost functions (Yang et al., 2023; Roth et al.), allowing models to infer navigation-relevant environmental structure more explicitly compared to direct policy learning. However, these methods still rely on sequential navigation data

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

and trajectory-based supervision, which requires costly trajectory collection and learns planning-oriented representations indirectly from demonstrated behaviors or optimization trajectories.

In light of these observations, we ask: *Can navigation-oriented planning structure be learned explicitly without relying on trajectory-centric supervision, while still retaining the reactive behavior of learned navigation systems and the structured reasoning of modular planners?* We posit that goal-conditioned cost-to-go representations provide a principled middle ground between these two paradigms. To this end, we propose Value-Guided Navigation (VGN) that learns traversability-aware and goal-conditioned cost-to-go representations directly from RGB observations. Rather than directly predicting navigation actions or trajectories, VGN explicitly models traversability-aware and goal-conditioned planning representations for downstream path generation. To achieve this, we introduce a scalable *supervision-generation pipeline* that combines vision foundation models with classical planning to transform static RGB imagery into dense traversability and goal-conditioned cost-to-go pseudo-labels directly in image space. In this way, abundant static visual data is converted into navigation-oriented supervision without expert demonstrations, robot interaction, or sequential navigation data, aligning with recent calls to transform large-scale world data into robot-usable supervision (Karcini et al., 2026). The resulting pseudo-labels are used to train the student model to predict planning representations directly from RGB observations. The student model uses a lightweight region-level *semantic distillation module* to align region representations derived from dense encoder features with text-grounded teacher embeddings, encouraging terrain-aware semantic structure in the learned representation. The distilled features then pass through a *traversability head* and a goal-conditioned *cost-to-go head* to predict dense traversability and planning representations. During inference, a lightweight planner uses the representations predicted from a single RGB observation and a projected image-space goal to generate navigation paths in image space. A lightweight controller then converts these waypoints into linear and angular velocity commands. Extensive experiments and real-world deployment on a Unitree Go2 robot demonstrate robust RGB-based traversability estimation and cost-to-go-guided local planning during closed-loop navigation.

2 Related Work

2.1 Modular Visual Navigation

Most modular visual navigation systems estimate traversability or navigability representations from visual observations and subsequently use downstream planners for trajectory generation. Early modular approaches relied on geometric and semantic terrain representations to construct traversability maps (Meng et al., 2023; Frey et al., 2024; Pushp et al., 2025). TerrainNet (Meng et al., 2023) and RoadRunner (Frey et al., 2024) jointly modeled semantic and geometric terrain understanding for off-road navigation, while systems such as POVNav (Pushp et al., 2025) converted Mask2former (Cheng et al., 2022) segmentation outputs directly into navigability costs. Although these approaches improve navigation safety in unstructured environments, they often rely on annotated terrain labels and manually designed costs. To reduce manual annotations, another line of work explored traversability learning from robot interactions and traversal experience (Gasparino et al., 2022; Sathyamoorthy et al.; Gasparino et al., 2024; Castro et al., 2023). TerraPN (Sathyamoorthy et al.) predicted dense traversability cost maps from RGB observations and robot motion signals, while WayFAST (Gasparino et al., 2022) focused on self-supervised traversable path prediction from robot experience. WayFASTER (Gasparino et al., 2024) further modeled traction-aware terrain understanding for off-road navigation. However, supervision is often limited to traversed trajectories, leading to sparse learning signals and limited generalization.

More recent approaches increasingly leverage vision foundation models (VFMs) for traversability estimation and navigation (Mattamala et al., 2025; Frey et al., 2023; Jung et al., 2024; Wang et al., 2025; Schreiber & Driggs-Campbell, 2025; Potnis et al., 2026). Methods such as Wild Visual Navigation (Mattamala et al., 2025), V-STRONG (Jung et al., 2024), and Genie (Wang et al., 2025) leverage DINO- and SAM-based (Oquab et al., 2023; Ravi et al., 2025) priors for traversability and terrain understanding. Chungus (Schreiber & Driggs-Campbell, 2025) additionally combines DINOv2 features with LiftUp-based upsampled visual representations (Suri et al., 2024) for navigation-aware terrain reasoning. Approaches such as CatNav (Potnis et al., 2026) and TRAIL (Jia et al., 2026) move beyond standard traversability estimation toward planning-oriented navigation representations, where CatNav leverages VLM-based cost-aware navigation, while TRAIL additionally uses LiDAR to learn implicit neural traversability representations for trajectory optimization. Our approach instead learns traversability-aware and goal-conditioned cost-to-go representations directly from single RGB observations for local image-space planning.

2.2 End-to-End Visual Navigation

End-to-end visual navigation methods aim to directly predict navigation actions, trajectories, or waypoints from onboard observations without relying on explicit mapping or multi-stage planning pipelines. One prominent direction is imitation learning, where navigation policies are learned from expert demonstrations (Bojarski et al., 2016; Pfeiffer et al., 2017; Bansal et al., 2018; Inglin et al., 2026; Lukas Busch et al., 2026). While such approaches can learn effective navigation behaviors, they often generalize poorly to unfamiliar environments and are limited by the quality and diversity of demonstrations. Recent methods such as LIMO (Inglin et al., 2026) attempt to address these limitations by combining expert demonstrations with geometric planner-generated trajectories for scalable supervision. Another major direction explores RL for end-to-end navigation through interaction-based learning (Zhang et al., 2016; Stachowicz et al., 2023; Xi et al., 2026; Huang et al., 2026; Seneviratne et al., 2025). RL enables navigation policies to be learned directly through interaction and simulation-based training, reducing dependence on demonstrations. However, low sampling efficiency and sparse rewards often make RL-based navigation computationally expensive, particularly for high-dimensional observations.

To address some limitations of imitation and reinforcement learning, iPlanner (Yang et al., 2023) introduced imperative learning via bilevel optimization over differentiable planning objectives without explicit trajectory demonstrations, while ViPlanner (Roth et al.) further extended this line of work with semantic-aware trajectory prediction. Recent advances increasingly explore transformer-, diffusion-, and generative-flow-based models for action and trajectory generation (Cai et al.; Zeng et al., 2025; Sridhar et al., 2024; Gode et al., 2024; Ren et al., 2025). NaviDiffusor (Zeng et al., 2025) investigates how to bridge classical cost-based planning and end-to-end learning through diffusion-guided path generation. Beyond point-goal navigation, NoMaD (Sridhar et al., 2024) and FlowNav (Gode et al., 2024) extend these frameworks toward goal-image-conditioned navigation, while recent methods further explore language-conditioned embodied navigation and action generation (Zhang et al., 2025; Hirose et al., 2024; Cheng et al., 2024; Kim et al., 2024; Hirose et al., 2025). Unlike goal-image-conditioned and vision-language approaches, our method focuses on RGB-based perception and local planning for point-goal navigation.

3 Methodology

This section presents our VGN framework for learning traversability-aware and goal-conditioned planning representations directly from monocular RGB observations. We first formulate the overall navigation problem and provide an overview of the proposed framework. Next, we describe the teacher-driven supervision generation process and the proposed planning representation learning framework. Finally, we describe downstream navigation using the learned representation.

3.1 Overview

Given a monocular RGB observation $I_t \in \mathbb{R}^{H\times W\times 3}$ and an image-space goal $\bar{g}_t$, the objective is to predict a dense traversability map $T_t \in \mathbb{R}^{H\times W}$ and a goal-conditioned cost-to-go representation $V_t \in \mathbb{R}^{H\times W}$ for local image-space planning. During deployment, $\bar{g}_t$ is obtained by projecting a global point goal g_t into the image frame using the current robot pose provided by an external localization system. A downstream planner $\mathcal{P}$ uses V_t built on top of T_t and the spatial goal to generate an image-space path $s_{1:N}$ of N waypoints, where $s_i = [u_i, v_i]$ denotes the pixel location of the i-th waypoint. To learn these representations, we formulate goal-conditioned cost-to-go prediction as a pseudo-supervised representation learning problem, as illustrated in Figure 1. During training, a

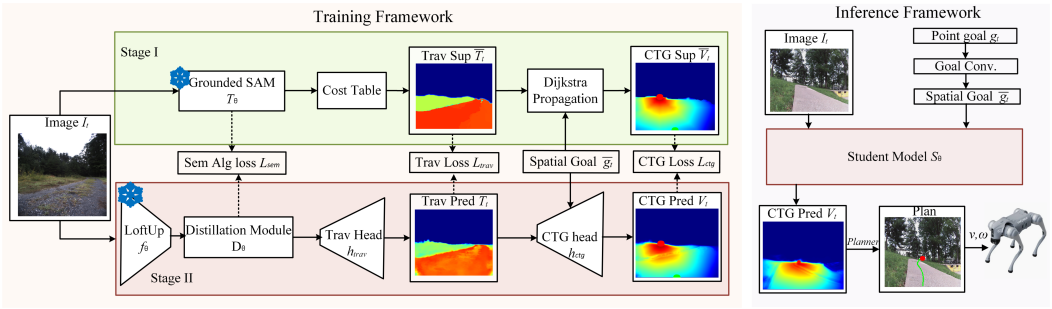

Figure 1: Overview of the proposed framework.

teacher model T_ϕ first generates dense traversability and goal-conditioned cost-to-go pseudo-labels

$(\bar{T}_t, \bar{V}_t)$ from the RGB observation I_t and a sampled image-space goal $\bar{g}_t$ (Sec. 3.2). A student model S_θ, consisting of a frozen LoftUp encoder f_θ (Huang et al., 2025), a semantic distillation module D_θ, and lightweight prediction heads h_{trav} and h_{ctg}, then learns the dense traversability map T_t and cost-to-go representation V_t by aligning region-level visual features with teacher region embeddings (Sec. 3.3). During inference, the student predicts T_t from the RGB observation and V_t conditioned on the projected image-space goal $\bar{g}_t$. These representations are subsequently used by the planner $\mathcal{P}$ for local path generation , followed by a lightweight controller for closed-loop navigation (Sec. 3.4).

3.2 Teacher-Driven Supervision Generation

Given an RGB observation I_t and a predefined set of terrain classes $\mathcal{C} = \{c_k\}_{k=1}^{K}$, our teacher model T_ϕ, based on GroundedSAM (Ren et al., 2024), extracts a set of text-grounded semantic regions $\mathcal{R}_t = \{(m_k, f_k, c_k)\}_{k=1}^{K}$, where m_k, f_k, and c_k denote the binary region mask, semantic region feature, and associated terrain category, of the k-th region, respectively. Specifically, GroundingDINO (Liu et al., 2024) first predicts text-conditioned region proposals, which are subsequently refined into dense object masks using SAM (Kirillov et al., 2023).

Each terrain category (i.e., road, grass, tree) is then assigned a traversability prior range $t_k \in [0, 1]$, where highly traversable regions are assigned higher traversability values while visually challenging terrain regions receive lower values. The traversability value of each region is estimated using the confidence score of the grounded prediction within its assigned traversability range. Since multiple text-grounded regions may spatially overlap, the dense traversability pseudo-label $\bar{T}_t$ is generated by averaging the traversability values over overlapping regions.

In the next stage, the resulting traversability field is used to construct goal-conditioned planning representations. We randomly sample image-space goals $\bar{g}_t$ from traversable regions and generate a dense cost-to-go pseudo-label $\bar{V}_t$ using Dijkstra-based (Dijkstra, 1959) cost-to-go propagation over the traversability field. Unlike trajectory-centric supervision that provides only sparse path-level guidance, the resulting dense cost-to-go representation enables the student model to learn the global planning structure of the environment across the entire image space. The propagation cost is defined as $\psi(i, j) = \alpha d(i, j) + \beta(1 - \bar{T}_t(j))^2$, where $d(i, j)$ denotes the spatial transition cost between neighboring pixels and $(1 - \bar{T}_t(j))^2$ penalizes propagation through low-traversability regions. In practice, $d(i, j)$ corresponds to the Euclidean step distance between neighboring pixels in the 8-connected image graph, encouraging spatially smooth propagation while biasing planning toward safer and more traversable regions. Together, the generated pseudo-labels $(\bar{T}_t, \bar{V}_t)$ provide the planning supervision for training the student network.

3.3 Planning Representation Learning

Given the RGB observation I_t, the student model S_θ first extracts dense visual features using a frozen LoftUp encoder f_θ, producing a high-resolution feature map $F_{raw} \in \mathbb{R}^{C \times H \times W}$. These features are then projected through a lightweight residual projection block of the semantic distillation module D_θ (Figure 2) to obtain projected features $F_{proj} \in \mathbb{R}^{C_p \times H \times W}$. From F_{proj}, we construct student region tokens $q_k \in \mathbb{R}^{C_p}$ through masked average pooling using the semantic masks m_k from the teacher model. We also construct the corresponding teacher semantic tokens $t_k \in \mathbb{R}^{C_p}$ from the grounded semantic embeddings f_k extracted by GroundingDINO, providing semantically grounded representations of the same physical regions. We then align the student region tokens with the corresponding teacher semantic tokens through lightweight cross-attention, $\hat{q}_k = \mathrm{Attn}(q_k, t_k, t_k)$, where the student tokens act as queries, and the teacher semantic tokens serve as keys and values in $\mathrm{Attn}(Q, K, V) = \mathrm{softmax}(QK^\top)V$, encouraging the student features to progressively reorganize the frozen visual representations toward semantically grounded terrain-aware structure. The refined semantic tokens are then injected back into the F_{proj} to produce a grounded representation F_{ground}. During training, we optimize the semantic distillation module D_θ using a region-level semantic alignment loss between the pooled grounded student features $\bar{f}_k$ from F_{ground} and the cor-

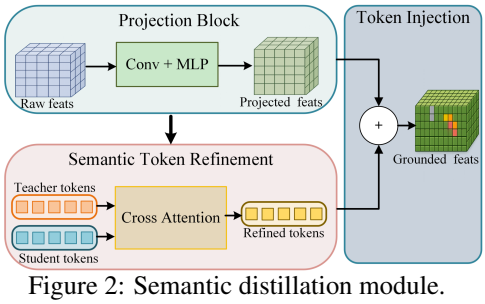

Figure 2: Semantic distillation module.

responding teacher embeddings f_k: $L_{sem} = 1 - \frac{f_k \cdot \bar{f}_k}{\|f_k\|\|f_k\|}$. Through backpropagation, the projected representation F_{proj} gradually acquires the distilled semantic grounding from the teacher-guided refinement process, enabling the student network to perform inference using only F_{proj}.

The grounded representation F_{ground} during training and the projected representation F_{proj} during inference are used for predicting a dense traversability map $T_t \in \mathbb{R}^{H\times W}$ through a lightweight convolutional prediction head h_{trav}. The cost-to-go head h_{ctg} (Figure 3) then receives the traversability map T_t together with a spatial goal representation $\bar{g}_t$ to generate the dense goal-conditioned cost-to-go representation $V_t \in \mathbb{R}^{H\times W}$. During inference, the target point goal g_t is first projected into the image space to construct the corresponding spatial goal representation $\bar{g}_t$. The h_{ctg} first constructs a Gaussian goal map G_t and normalized unit-direction field U_t from $\bar{g}_t$, where G_t provides a dense spatial goal prior and U_t encodes pixel-wise directional cues toward the target location. In parallel, normalized coordinate features are fused with the traversability-aware features to preserve spatial layout information. The resulting goal-conditioned representation $F_{goal} = [G_t; U_t]$ is then used to modulate the traversability-aware features F_{trav} through FiLM conditioning: $F_{plan} = \gamma(F_{goal}) \odot F_{trav} + \beta(F_{goal})$, where $\gamma(\cdot)$ and $\beta(\cdot)$ denote learned modulation functions. This conditioning modulates the traversability-aware features into goal-aware planning representations by modulating the feature maps according to the spatial goal location and directional context. While FiLM conditioning provides local goal-aware feature modulation, lightweight cross-attention refinement is further applied between F_{plan} and the goal-conditioned features F_{goal} to propagate planning-aware contextual interactions across spatial regions, producing the contextual planning representation F_{ctx}. Traversability-aware cost fusion further reintegrates the traversability structure T_t into F_{ctx} to preserve terrain-aware planning cues, followed by residual refinement to obtain the final planning representation F_{final}. Finally, a lightweight 1×1 convolutional prediction layer produces the dense cost-to-go representation V_t. During training, the traversability and cost-to-go predictions are optimized using Huber loss: $L_{trav} = \text{Huber}(T_t, \bar{T}_t)$ and $L_{ctg} = \text{Huber}(V_t, \bar{V}_t)$. The overall training objective is defined as $L = L_{trav} + \lambda_1 L_{ctg} + \lambda_2 L_{sem}$.

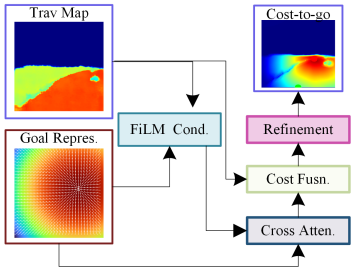

Figure 3: Cost-to-go head.

3.4 Navigation with the Learned Representation

Given the predicted traversability map T_t, goal-conditioned cost-to-go representation V_t, and projected goal $\bar{g}_t$, a downstream planner $\mathcal{P}$ generates an image-space waypoint sequence from the bottom-center image location toward the projected goal. We use "value" in the optimal-control sense of a goal-conditioned cost-to-go function, where lower values indicate lower predicted navigation cost. The traversability map defines the feasible navigation region, while the learned value representation provides goal-directed planning guidance. We additionally derive an obstacle-clearance cost C_t^{clr} from T_t, and define the shared spatial planning cost as

$$\Phi_t(x) = \lambda_V V_t(x) + \lambda_C C_t^{\text{clr}}(x). \tag{1}$$

The planner uses this representation to generate collision-aware waypoints toward $\bar{g}_t$. We apply this planning representation across multiple downstream planning strategies, with the navigation planner details provided in the Appendix E. A look-ahead target point is then selected from the planned path and a lightweight controller converts it into control commands. Specifically, given the target pixel location $q_t = (x_t, y_t)$, the controller computes the normalized image-space tracking error e_x relative to the image center and predicts angular velocity through proportional heading control, $\omega = -k_\omega e_x$, while the forward velocity v is adaptively scaled according to the turning magnitude to preserve stable motion during navigation. Global pose monitoring is further used for near-goal heading stabilization and stopping logic to ensure smooth goal reaching.

4 Experiments and Results

Experimental Setup. To evaluate our method, we aim to address the following questions: (Q1) Does VGN generalize across unseen environments? (Q2) Do high-resolution visual features improve planning representations? (Q3) How effectively does VGN perform closed-loop navigation

in simulated environments? (Q4) Can VGN be deployed for real-world robot navigation? To answer these questions, we train the proposed framework on the RUGD (Wigness et al., 2019), RELLIS (Jiang et al., 2021), and Cityscapes (Cordts et al., 2016) datasets and evaluate across multiple unseen outdoor datasets, simulated navigation environments, and real-world deployment scenarios using a Unitree Go2 quadruped robot equipped with an onboard Jetson Orin.

Cross-Environment Generalization. To evaluate cross-environment generalization, we train the proposed framework on the RUGD dataset and evaluate it on the validation sets of RUGD, RELLIS, GOOSE (Mortimer et al., 2024), NaviTrace (Windecker et al., 2025), and Cityscapes without target-domain fine-tuning. We additionally train VGN on RELLIS and evaluate it across the same set of datasets, providing a complementary assessment of cross-domain transfer. NaviTrace and Cityscapes introduce larger appearance and environmental shifts from the off-road training domains, allowing us to assess generalization beyond visually similar terrain settings. For traversability evaluation, we report MAE, RMSE, IoU@0.5, and Pearson correlation. For goal-conditioned cost-to-go evaluation, we report MAE, RMSE, Pearson correlation, and Gradient Cosine Similarity. The quantitative cross-environment evaluation results are summarized in Table 1. Figure 4 further visualizes representative traversability and goal-conditioned cost-to-go predictions across the three environments. Across the evaluated domains, the results demonstrate meaningful traversability and goal-conditioned cost-to-go structure, although performance depends on the training distribution. The RUGD-trained model transfers relatively well, whereas the RELLIS-trained model shows larger cross-domain degradation. Overall, these results indicate non-trivial cross-domain transfer while highlighting its source dependence (Q1). To complement the teacher-derived dense evaluation, we further evaluate traversability using the human pairwise annotations and Human Disagreement Rate (HDR) protocol used by CHUNGUS (Schreiber & Driggs-Campbell, 2025). HDR measures the rate at which the relative traversability predicted for a pair of annotated pixels disagrees with the corresponding human judgment. For a threshold $\tau \in \{0.1, 0.25, 0.5\}$, one pixel is considered more or less traversable than the other only when their predicted traversability scores differ by more than τ; otherwise, they are treated as equally traversable. VGN is trained only on RUGD using teacher-generated pseudo-labels based on semantic traversability priors, without pairwise human supervision, and is evaluated zero-shot on the unseen CHUNGUS real-world images (Q1). In contrast, CHUNGUS and W-RIZZ are trained directly using sparse human pairwise traversability labels. Despite this different supervision strategy, VGN remains competitive and achieves the lowest HDR@0.5, suggesting stronger agreement with human judgments under a coarser traversability distinction, as shown in Table 2.

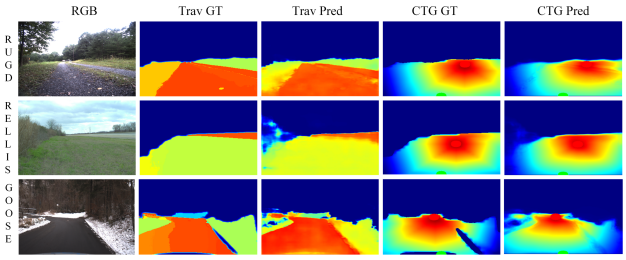

Figure 4: Qualitative cross-environment generalization results across datasets.

Table 1: Cross-environment generalization results.

Train→Test	Traversability Metrics				Cost-to-go Metrics			
	MAE↓	RMSE↓	IoU↑	Corr↑	MAE↓	RMSE↓	Corr↑	GradCos↑
RUGD→RUGD	0.037	0.095	0.933	0.961	0.038	0.101	0.952	0.365
RUGD→RELLIS	0.060	0.136	0.880	0.857	0.048	0.115	0.968	0.383
RUGD→GOOSE	0.073	0.165	0.816	0.920	0.061	0.134	0.864	0.283
RUGD→NaviTrace	0.098	0.210	0.764	0.825	0.032	0.179	0.775	0.282
RUGD→Cityscapes	0.050	0.134	0.915	0.938	0.043	0.111	0.920	0.371
RELLIS→RELLIS	0.047	0.116	0.903	0.939	0.039	0.106	0.932	0.408
RELLIS→RUGD	0.109	0.204	0.794	0.806	0.060	0.148	0.878	0.305
RELLIS→GOOSE	0.127	0.222	0.730	0.797	0.073	0.157	0.818	0.267
RELLIS→NaviTrace	0.185	0.281	0.651	0.657	0.108	0.207	0.715	0.251
RELLIS→Cityscapes	0.200	0.289	0.712	0.723	0.080	0.166	0.826	0.307

Effect of High-Resolution Features on Planning Representations. To analyze the effect of high-resolution visual features on planning representations, we evaluate our framework using three frozen visual backbones: ResNet50, DINO, and LoftUp. For each scenario, the teacher model first generates traversability and cost-to-go (CTG) representations, and then the planner is used

Table 2: Independent Traversability Evaluation.

Method	HDR@0.1↓	HDR@0.25↓	HDR@0.5↓
WVN	0.494	0.428	0.367
W-RIZZ	0.269	<u>0.218</u>	0.291
Chungus	**0.159**	**0.126**	0.287
VGN	<u>0.217</u>	0.232	**0.261**

to obtain the reference path. We then replace the frozen student backbone with each visual encoder and generate planned paths under the identical planning setup.

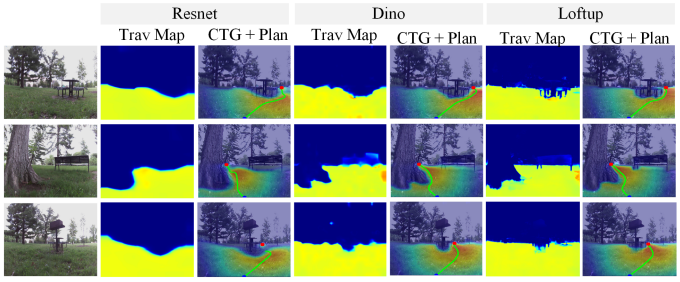

Figure 5: Qualitative comparison of planning representations generated using different frozen visual backbones.

The evaluated scenarios are unseen during training (Q1) and contain cluttered outdoor environments that require stronger spatial reasoning and boundary understanding. Figure 5 shows qualitative comparisons on three representative scenarios. Compared to ResNet, both DINO and LoftUp produce more spatially consistent traversability and planning representations, resulting in smoother and more goal-consistent paths (Q2). Quantitative results in Table 3 further show that stronger high-resolution visual features improve open-loop planning quality, as reflected by several metrics including path efficiency, endpoint error, path deviation, and path smoothness (Q2). All planning metrics are evaluated in image space by comparing planner-generated paths from the student-predicted dense cost-to-go representations against teacher-generated reference paths under the identical planning setup. As the LoftUp-based variant provides the strongest spatial consistency and planning representation quality, we therefore use it for all subsequent closed-loop navigation experiments.

Table 3: Open-loop planning evaluation using different frozen visual backbones.

Method	Path Efficiency↑	End Err↓	Path Dev↓	Smooth↓
ResNet	0.5975	9.4861	7.4111	0.0225
DINO	0.9132	4.3579	**4.3392**	**0.0200**
LoftUp	**0.9217**	**4.1455**	5.3434	0.0201

Closed Loop Navigation in Simulation To evaluate closed-loop navigation performance in simulation, we use the CMU Autonomous Exploration Development Simulator (Zhang et al., 2020) and generate 100 randomized episodes with unique start–goal configurations for the *Forest* and *City* environments. We conduct two complementary evaluation protocols. First, we compare VGN against iPlanner, ViPlanner, and POVNav using their complete navigation stacks in a fully closed-loop setting on unseen simulated environments. Within the VGN stack, we evaluate three downstream planning strategies over the learned CTG representation: graph-based search, direct gradient rollout, and gradient-guided sampling (Q1) (Q3). Second, to isolate the effect of traversability estimation, we evaluate the traversability predictions from WVN, W-RIZZ, CHUNGUS, and VGN using the same navigation module across identical navigation episodes. Table 4 reports the success rate and Success weighted by normalized inverse Path Length (SPL), together with 95% confidence intervals. An episode is considered successful if the robot reaches the target goal within a distance threshold of 0.5 m without collision. SPL is computed only over successful episodes to measure path efficiency relative to the shortest start–goal distance.

Table 4: Closed-loop navigation performance across Forest and City simulation environments. Values in brackets denote 95% confidence intervals.

Evaluation Protocol	**Method**	**Forest**		**City**	
		Success ↑	**SPL ↑**	**Success ↑**	**SPL ↑**
Complete Navigation Stack	iPlanner	64.0% [54.2, 72.7]	0.639 [0.548, 0.730]	65.0% [55.3, 73.6]	0.649 [0.558, 0.739]
	ViPlanner	55.0% [45.2, 64.4]	0.546 [0.447, 0.644]	70.0% [60.4, 78.1]	0.699 [0.609, 0.789]
	POVNav	55.0% [45.2, 64.4]	0.465 [0.381, 0.549]	21.0% [14.2, 30.0]	0.173 [0.109, 0.243]
	VGN (graph-search)	**80.0% [71.1, 86.7]**	0.682 [0.610, 0.751]	68.0% [58.3, 76.3]	0.659 [0.569, 0.746]
	VGN (graph-search w/o heuristic)	**80.0% [71.1, 86.7]**	0.818 [0.786, 0.850]	62.0% [52.2, 70.9]	0.922 [0.885, 0.959]
	VGN (direct gradient)	68.0% [58.3, 76.3]	0.853 [0.818, 0.888]	**71.0% [61.5, 79.0]**	0.957 [0.932, 0.981]
	VGN (gradient-based sampling)	65.0% [55.3, 73.6]	**0.924 [0.904, 0.944]**	68.0% [58.3, 76.3]	**0.989 [0.979, 0.998]**
Traversability Estimator + Fixed Navigation Module	WVN	2.0% [0.6, 7.0]	0.020 [0.000, 0.049]	49.0% [39.4, 58.7]	0.471 [0.377, 0.567]
	W-RIZZ	64.0% [54.2, 72.7]	0.631 [0.537, 0.723]	11.0% [6.3, 18.6]	0.109 [0.050, 0.170]
	CHUNGUS	70.0% [60.4, 78.1]	0.697 [0.607, 0.786]	63.0% [53.2, 71.8]	0.617 [0.522, 0.710]
	VGN (RUGD)	72.0% [62.5, 79.9]	0.717 [0.628, 0.806]	57.0% [47.2, 66.3]	0.568 [0.469, 0.666]
	VGN (Cityscapes)	**81.0% [72.2, 87.5]**	**0.995 [0.992, 0.998]**	**73.0% [63.6, 80.7]**	**0.992 [0.983, 1.000]**

For the complete navigation-stack evaluation, VGN consistently achieves strong closed-loop navigation performance. In Forest, all VGN variants outperform the baselines, with the graph-based VGN variant achieving 80% success, a substantial 16 percentage-point improvement over the strongest baseline (64%). In City, the direct gradient rollout achieves 71% success, slightly surpassing ViPlan-

ner (70%), while the graph-based and gradient-guided sampling variants remain close at 68%. These results highlight the versatility of the learned CTG representation across different downstream planning mechanisms. Notably, the direct gradient variant shows that the learned CTG can directly drive navigation through gradient rollout, without graph search or trajectory sampling, while still outperforming all evaluated baselines in both environments. With a fixed navigation stack, VGN achieves 72% success in Forest and 57% in City, outperforming the other traversability estimators in Forest while remaining competitive with CHUNGUS in City. To further examine how the training-data distribution affects traversability estimation, we also train VGN on Cityscapes and evaluate its learned traversability representation under the fixed navigation stack. The Cityscapes-trained traversability representation improves to 81% success and 0.995 SPL in Forest, and 73% success and 0.992 SPL in City, achieving the best performance among the evaluated traversability estimators in both environments. Overall, VGN demonstrates strong closed-loop navigation performance across both evaluation protocols (Q3). To isolate the contribution of the learned cost-to-go representation, we replace the CTG with continuous traversability while keeping the same graph-based image-space planner and all other planning components unchanged. As shown in Table 5, CTG-guided planning improves the success rate from 64% to 68% in City, while maintaining the same success rate in Forest. Since the downstream graph-based planner already searches over the current image-space grid, traversability alone can be sufficient in broad, open regions with large free corridors, as observed in Forest. In more constrained settings, however, locally traversable regions may differ in their downstream connectivity to the goal. The modest improvement in City suggests that CTG can capture this downstream route structure beyond local traversability and provide additional goal-directed guidance. Beyond this, Table 6 further compares different combinations of teacher- and student-predicted traversability and cost-to-go representations. The learned traversability representation provides a smoother and more planner-compatible

Table 5: Effect of CTG on navigation.

	Forest		City	
Input	**Success**	**SPL**	**Success**	**SPL**
Traversability	80.0%	**0.845**	64.0%	**0.614**
Learned CTG	**80.0%**	0.682	**68.0%**	0.659

Table 6: Teacher–student ablation for closed-loop navigation and inference efficiency.

Traversability	**CTG**	**Forest**		**City**		**Runtime (RTX 4090)**			
		Success↑	**SPL↑**	**Success↑**	**SPL↑**	**Trav. (ms)↓**	**CTG (ms)↓**	**Total (ms)↓**	**FPS↑**
Teacher	Teacher	67%	0.835	48%	0.920	461.0 ± 21.1	12.8 ± 5.0	473.8 ± 20.7	2.11
VGN	Teacher	**82%**	0.844	66%	0.636	–	–	–	–
Teacher	VGN	63%	**0.854**	66%	0.640	–	–	–	–
VGN	VGN	80%	0.682	**68%**	**0.659**	$\mathbf{35.2 \pm 3.5}$	$\mathbf{7.5 \pm 1.3}$	$\mathbf{42.7 \pm 3.8}$	**23.44**

estimate than the raw teacher output. The learned CTG preserves comparable goal-conditioned planning behavior while replacing deterministic Dijkstra-based CTG recomputation with a learned prediction. Although the teacher Dijkstra-based CTG is already efficiently computed on an $8\times$ downsampled grid (64×64), VGN distills this cost structure into a dense 224×224 representation. Together, the learned traversability and CTG enable a unified and efficiently deployable navigation pipeline. Figure 6 further shows representative closed-loop navigation episodes in the Forest and

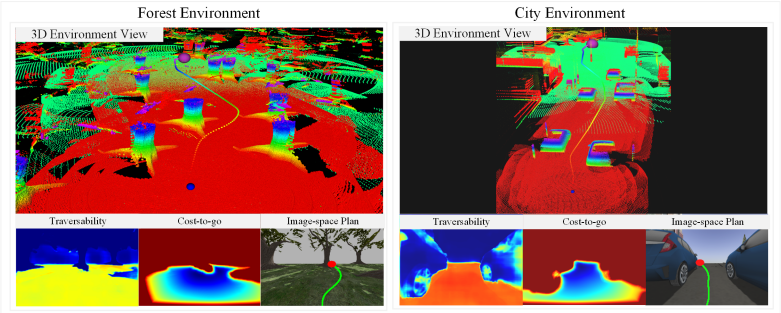

Figure 6: Closed-loop navigation demonstration in simulation environments.

City environments, including the executed robot trajectory and a representative timestep showing the predicted traversability, goal-conditioned cost-to-go, and image-space plan. Moreover, we report motion smoothness in Figure 7 using mean jerk and mean yaw rate, where lower values indicate smoother motion. VGN achieves the lowest values for both metrics under the complete navigation stack, while under the fixed planner, Cityscapes-trained VGN achieves the lowest mean jerk and CHUNGUS the lowest mean yaw rate across both environments. Together, all these results demonstrate that VGN achieves effective closed-loop navigation while maintaining smooth motion across

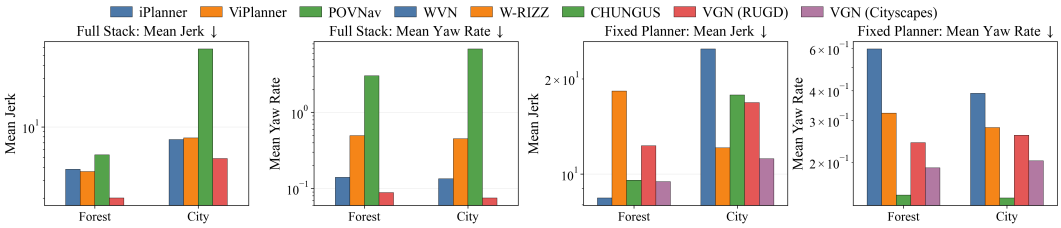

Figure 7: Motion smoothness comparison of different methods.

diverse simulated environments (Q1) (Q3). Moreover, the learned CTG can be integrated with different downstream planning strategies, highlighting its planner-agnostic utility.

Real-World Robot Navigation Demonstration. To evaluate real-world deployment capability, we deploy VGN on a Unitree Go2 quadruped robot equipped with an onboard Jetson Orin across diverse navigation scenarios (Q1, Q4). We evaluate the system under different terrain layouts, visual conditions, and illumination settings, including both daytime and evening navigation. Figure 13 in appendix G shows representative closed-loop deployments. Each example includes a third-person view together with a representative timestep showing the predicted traversability map, goal-conditioned cost-to-go representation, and corresponding image-space plan. Qualitative videos of the real-world deployments are available here. Moreover, Table 7 summarizes the onboard computational cost of VGN, including the latency of each major component and the end-to-end RGB-to-local-motion plan pipeline.

Table 7: Onboard runtime during real-robot deployment.

Component	**Latency (ms, mean ± std)**	**Deployment Metric**	**Value**
DINOv2 + LoftUp	482.0 ± 6.2	**RGB-to-Local Motion Plan**	$\mathbf{716.3 \pm 13.8}$ ms
Semantic Module + Traversability Head	51.9 ± 1.8	95th-percentile latency	733.9 ms
Goal-conditioned CTG Head	111.6 ± 2.6	Planning frequency	1.40 Hz
Graph-based Planning	57.7 ± 6.6	Peak GPU memory	906.1 MB

The system operates at approximately 1.40 Hz with a peak GPU memory usage of 906.1 MB, demonstrating practical onboard execution on the real robot. Together, these demonstrations highlight the practical deployment of VGN for closed-loop navigation (Q4) across diverse real-world conditions without environment-specific retraining.

5 Limitations

Our traversability and planning representations are learned solely on monocular RGB observations and do not explicitly model geometric depth or 3D scene structure, which can sometimes lead to ambiguous traversability predictions in visually challenging regions such as tall grass or uneven terrain (Figure 11 left in appendix D). Moreover, due to the lack of explicit depth and scene geometry reasoning, projected image-space goals may occasionally fall onto foreground obstacles or partially occluded regions. While the proposed replanning mechanism can mostly recover by progressively updating local subgoals during navigation (Figure 12 in appendix D), the framework may still struggle in certain cases involving very nearby or large obstacles that heavily occlude the final goal direction. In addition, planning representation learning currently lacks explicit consideration of robot footprint and leg-level terrain interaction. As a result, even after generating a visually feasible path, individual legs may still interact with local terrain structures such as tree roots or cluttered obstacles while following the planned path (Figure 11 right in appendix D). These limitations motivate future directions toward depth-aware planning representations and predictive world modeling for future cost-to-go reasoning and embodiment-aware navigation.

6 Conclusions

In this work, we introduced VGN, a modular visual navigation framework that learns traversability-aware and goal-conditioned planning representations directly from RGB observations. The proposed framework learns planning-oriented visual representations that can support multiple downstream navigation strategies across diverse outdoor environments. Through experiments across multiple outdoor datasets and real-world deployment on the Unitree Go2, we demonstrated effective RGB-based traversability estimation and cost-to-go-guided local planning without relying on trajectory-centric supervision. We believe the proposed formulation provides a promising direction toward visual navigation systems that explicitly learn dense goal-conditioned planning structure from monocular RGB observations. Furthermore, the framework highlights a meaningful middle ground between classical modular navigation pipelines and end-to-end learned navigation systems.

AI USE STATEMENT

In this work, we used generative AI tools for assisting method implementation. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning or reformatting datasets, or interpreting results, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating or debugging software code, brainstorming, drafting parts of the paper based on author-provided technical content, and refining the paper to improve presentation. We have reviewed all AI-assisted work. AI-assisted code was reviewed and tested for correctness by the authors, and AI-assisted text was reviewed and revised to ensure technical accuracy and consistency with the authors' intended meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or the collection of private or sensitive data. All datasets used in this study are publicly available research datasets and were used in accordance with their respective terms and licenses. We are not aware of any significant ethical, privacy, security, legal, or conflict-of-interest concerns arising from this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide anonymized source code as part of the supplementary materials and also at this link. The accompanying README provides instructions for environment setup, model training, evaluation, and reproducing the experiments reported in this paper. We also provide the navigation stack and associated configurations required to reproduce the navigation experiments. Additional implementation and experimental details are provided in the main paper and appendix.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

ACKNOWLEDGMENTS

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

W Cai, J Peng, Y Yang, Y Zhang, M Wei, H Wang, Y Chen, T Wang, and J Pang. Navdp: Learning sim-to-real navigation diffusion policy with privileged information guidance. arxiv 2025. *arXiv preprint arXiv:2505.08712*.

Chao Cao, Hongbiao Zhu, Fan Yang, Yukun Xia, Howie Choset, Jean Oh, and Ji Zhang. Autonomous exploration development environment and the planning algorithms. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 8921–8928. IEEE, 2022.

Mateo Guaman Castro, Samuel Triest, Wenshan Wang, Jason M Gregory, Felix Sanchez, John G Rogers, and Sebastian Scherer. How does it feel? self-supervised costmap learning for off-road vehicle traversability. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 931–938. IEEE, 2023.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. Navila: Legged robot vision-language-action model for navigation. *arXiv preprint arXiv:2412.04453*, 2024.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1290–1299, 2022.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 (1):269–271, 1959.

Jonas Frey, Matias Mattamala, Nived Chebrolu, Cesar Cadena, Maurice Fallon, and Marco Hutter. Fast traversability estimation for wild visual navigation. *arXiv preprint arXiv:2305.08510*, 2023.

Jonas Frey, Manthan Patel, Deegan Atha, Julian Nubert, David Fan, Ali Agha, Curtis Padgett, Patrick Spieler, Marco Hutter, and Shehryar Khattak. Roadrunner—learning traversability estimation for autonomous off-road driving. *IEEE Transactions on Field Robotics*, 1:192–212, 2024.

Mateus V Gasparino, Arun N Sivakumar, Yixiao Liu, Andres EB Velasquez, Vitor AH Higuti, John Rogers, Huy Tran, and Girish Chowdhary. Wayfast: Navigation with predictive traversability in the field. *IEEE Robotics and Automation Letters*, 7(4):10651–10658, 2022.

Mateus V Gasparino, Arun N Sivakumar, and Girish Chowdhary. Wayfaster: a self-supervised traversability prediction for increased navigation awareness. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8486–8492. IEEE, 2024.

Samiran Gode, Abhijeet Nayak, and Wolfram Burgard. Flownav: Learning efficient navigation policies via conditional flow matching. In *2nd CoRL Workshop on Learning Effective Abstractions for Planning*, 2024.

Noriaki Hirose, Catherine Glossop, Ajay Sridhar, Dhruv Shah, Oier Mees, and Sergey Levine. Lelan: Learning a language-conditioned navigation policy from in-the-wild videos. *arXiv preprint arXiv:2410.03603*, 2024.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Omnivla: An omni-modal vision-language-action model for robot navigation. *arXiv preprint arXiv:2509.19480*, 2025.

Fan Huang, HaiMing Mou, and QingDu Li. Tnavrl: Cross-modal transformer for humanoid visual navigation. *IEEE Robotics and Automation Letters*, 2026.

Haiwen Huang, Anpei Chen, Volodymyr Havrylov, Andreas Geiger, and Dan Zhang. Loftup: Learning a coordinate-based feature upsampler for vision foundation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9913–9923, 2025.

Yves Inglin, Jonas Frey, Changan Chen, and Marco Hutter. Less is more: Scalable visual navigation from limited data. *arXiv preprint arXiv:2601.17815*, 2026.

Yixuan Jia, Qingyuan Li, and Jonathan P How. Off-road navigation via implicit neural representation of terrain traversability. *IEEE Robotics and Automation Letters*, 2026.

Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. Rellis-3d dataset: Data, benchmarks and analysis. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 1110–1116. IEEE, 2021.

Sanghun Jung, JoonHo Lee, Xiangyun Meng, Byron Boots, and Alexander Lambert. V-strong: Visual self-supervised traversability learning for off-road navigation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1766–1773. IEEE, 2024.

Elis Karcini, Faisal Mehrban, Quang Nguyen, Mac Schwager, Arash Ajoudani, Cesar Cadena, Jan Peters, Marco Hutter, and Haitham Bou-Ammar. Robots need more than vla and world models. *arXiv preprint arXiv:2606.06556*, 2026.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, October 2023.

Mihir Kulkarni and Kostas Alexis. Reinforcement learning for collision-free flight exploiting deep collision encoding. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15781–15788. IEEE, 2024.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pp. 38–55. Springer, 2024.

Finn Lukas Busch, Matti Vahs, Quantao Yang, Jesús Gerardo Ortega Peimbert, Yixi Cai, Jana Tumova, and Olov Andersson. Learning to localize reference trajectories in image-space for visual navigation. *arXiv e-prints*, pp. arXiv–2602, 2026.

Matias Mattamala, Jonas Frey, Piotr Libera, Nived Chebrolu, Georg Martius, Cesar Cadena, Marco Hutter, and Maurice Fallon. Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision. *Autonomous Robots*, 49(3):19, 2025.

Xiangyun Meng, Nathan Hatch, Alexander Lambert, Anqi Li, Nolan Wagener, Matthew Schmittle, JoonHo Lee, Wentao Yuan, Zoey Chen, Samuel Deng, et al. Terrainnet: Visual modeling of complex terrain for high-speed, off-road navigation. *arXiv preprint arXiv:2303.15771*, 2023.

Peter Mortimer, Raphael Hagmanns, Miguel Granero, Thorsten Luettel, Janko Petereit, and Hans-Joachim Wuensche. The goose dataset for perception in unstructured environments. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14838–14844. IEEE, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1527–1533. IEEE, 2017.

Aditya Potnis, Francisco Affonso, Shreya Gummadi, Naveen Kumar Uppalapati, and Girish Chowdhary. Catnav: Cached vision-language traversability for efficient zero-shot robot navigation. *arXiv preprint arXiv:2603.22800*, 2026.

Durgakant Pushp, Weizhe Chen, Zheng Chen, Chaomin Luo, Jason M Gregory, and Lantao Liu. Navigating the wild: Pareto-optimal visual decision-making in image space. *The International Journal of Robotics Research*, pp. 02783649251399686, 2025.

Prithvi Jai Ramesh, Kaustav Chanda, Krishna Vinod, Joseph Raj Vishal, Yezhou Yang, and Bharatesh Chakravarthi. enavi: Event-based imitation policies for low-light indoor mobile robot navigation. *arXiv preprint arXiv:2603.14397*, 2026.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pp. 28085–28128, 2025.

Hao Ren, Yiming Zeng, Zetong Bi, Zhaoliang Wan, Junlong Huang, and Hui Cheng. Prior does matter: Visual navigation via denoising diffusion bridge models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12100–12110, 2025.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

Pascal Roth, Julian Nubert, Fan Yang, Mayank Mittal, and Marco Hutter. Viplanner: Visual semantic imperative learning for local navigation, 2024. *URL https://arxiv. org/abs/2310*, 982.

Adarsh Jagan Sathyamoorthy, Kasun Weerakoon, Tianrui Guan, Jing Liang, and Dinesh Manocha. Terrapn: Unstructured terrain navigation using online self-supervised learning. in 2022 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7197–7204.

Andre Schreiber and Katherine Driggs-Campbell. Do you know the way? human-in-the-loop understanding for fast traversability estimation in mobile robotics. *IEEE Robotics and Automation Letters*, 2025.

Gershom Seneviratne, Jianyu An, Sahire Ellahy, Kasun Weerakoon, Mohamed Bashir Elnoor, Jonathan Deepak Kannan, Amogha Thalihalla Sunil, and Dinesh Manocha. Halo: Human preference aligned offline reward learning for robot navigation. *arXiv preprint arXiv:2508.01539*, 2025.

Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7226–7233. IEEE, 2023a.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning (CoRL)*, 2023b. URL https://arxiv.org/abs/2306.14846.

Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 63–70. IEEE, 2024.

Kyle Stachowicz, Dhruv Shah, Arjun Bhorkar, Ilya Kostrikov, and Sergey Levine. Fastrlap: A system for learning high-speed driving via deep rl and autonomous practicing. In *Conference on Robot Learning*, pp. 3100–3111. PMLR, 2023.

Saksham Suri, Matthew Walmer, Kamal Gupta, and Abhinav Shrivastava. Lift: A surprisingly simple lightweight feature transform for dense vit descriptors. In *European Conference on Computer Vision*, pp. 110–128. Springer, 2024.

Joanne Truong, Max Rudolph, Naoki Harrison Yokoyama, Sonia Chernova, Dhruv Batra, and Akshara Rai. Rethinking sim2real: Lower fidelity simulation leads to higher sim2real transfer in navigation. In *conference on Robot Learning*, pp. 859–870. PMLR, 2023.

Hongwu Wang, Shiwei Lian, and Feitian Zhang. Collision-aware object-goal visual navigation via two-stage deep reinforcement learning, 2026. URL https://arxiv.org/abs/2502.13498.

Jiaming Wang, Diwen Liu, Jizhuo Chen, Jiaxuan Da, Nuowen Qian, Minh Man Tram, and Harold Soh. Genie: A generalizable navigation system for in-the-wild environments. *IEEE Robotics and Automation Letters*, 2025.

Lorenz Wellhausen and Marco Hutter. Rough terrain navigation for legged robots using reachability planning and template learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6914–6921. IEEE, 2021.

Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A rugd dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5000–5007. IEEE, 2019.

Tim Windecker, Manthan Patel, Moritz Reuss, Richard Schwarzkopf, Cesar Cadena, Rudolf Lioutikov, Marco Hutter, and Jonas Frey. Navitrace: Evaluating embodied navigation of vision-language models. *arXiv preprint arXiv:2510.26909*, 2025.

Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6750–6759, 2019.

Haoyu Xi, Mingao Tan, Xinming Zhang, Siwei Cheng, Shanze Wang, Yin Gu, Xiaoyu Shen, and Wei Zhang. Cerlp: A cross-embodiment robot local planning framework for visual navigation. *arXiv preprint arXiv:2603.19602*, 2026.

Fan Yang, Chen Wang, Cesar Cadena, and Marco Hutter. iplanner: Imperative path planning. *arXiv preprint arXiv:2302.11434*, 2023.

Yiming Zeng, Hao Ren, Shuhang Wang, Junlong Huang, and Hui Cheng. Navidiffusor: Cost-guided diffusion model for visual navigation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11994–12001. IEEE, 2025.

Arthur Zhang, Xiangyun Meng, Luca Calliari, Dong-Ki Kim, Shayegan Omidshafiei, Joydeep Biswas, Ali Agha, and Amirreza Shaban. Ventura: Adapting image diffusion models for unified task conditioned navigation. *arXiv preprint arXiv:2510.01388*, 2025.

Ji Zhang, Chen Hu, Rushat Gupta Chadha, and Sanjiv Singh. Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation. *Journal of Field Robotics*, 37(8):1300–1313, 2020.

Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 528–535. IEEE, 2016.

A Teacher Supervision Generation Details

A.1 Teacher Traversability Supervision

We use a predefined set of terrain-aware semantic phrases to generate GroundingDINO region proposals and refine them into dense segmentation masks using SAM. We generate region proposals using a box confidence threshold of 0.15 and a text confidence threshold of 0.15. We assign manually defined traversability prior ranges to each detected terrain category and estimate the final traversability value of each region using $T = a + p(b - a)$, where p denotes the GroundingDINO confidence score and $[a, b]$ represents the traversability prior range assigned to the detected terrain category. Table 8 summarizes the semantic traversability prior ranges used for teacher supervision generation. We aggregate overlapping semantic masks through weighted averaging to obtain the final dense traversability supervision map.

Table 8: Semantic traversability prior ranges used for pseudo traversability supervision generation.

Semantic Phrase	Traversability Range
road, pavement, sidewalk, concrete	0.75–1.0
grass, lawn, soil, gravel	0.5–0.75
mulch, sand, dirt, snow, mud	0.3–0.5
tree, rock, boulder, building, fence, pole, car, person, water, sky, bush, cloud	0.0

A.2 Teacher Cost-to-Go Supervision

We generate dense cost-to-go supervision using traversability-aware Dijkstra propagation over the pseudo traversability map. We sample image-space goals from traversable regions within the upper

60% of the image to encourage longer-range forward planning behaviors while avoiding trivial near-start goals close to the robot location at the bottom-center image region. Table 9 summarizes the parameters used for teacher cost-to-go supervision generation. We normalize the generated dense

Table 9: Teacher cost-to-go generation parameters.

Parameter	Value
Traversability threshold	0.3
Obstacle inflation radius	2
Cost map scale	8
Propagation weight α	1.0
Traversability penalty weight β	4.0
Number of sampled goals	1

cost-to-go supervision into the range $[0, 1]$ before using it for student training. Figure 8 shows representative teacher-generated traversability and cost-to-go supervision examples.

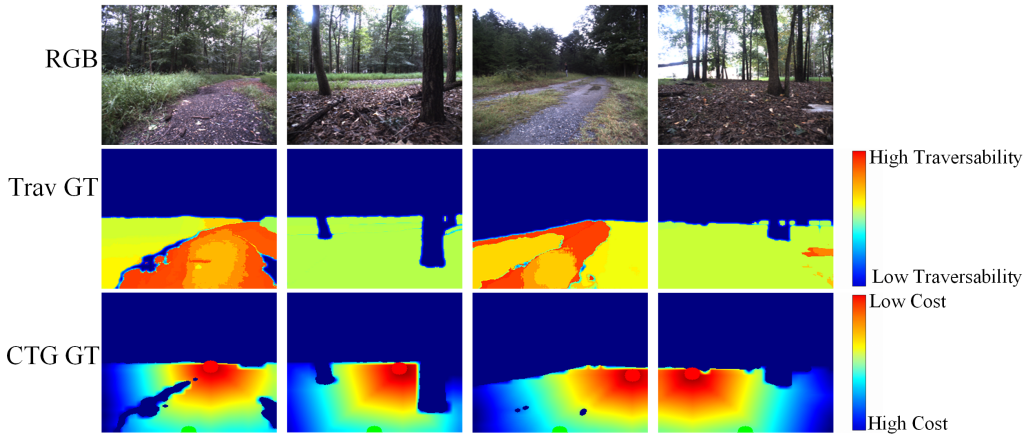

Figure 8: Representative teacher-generated traversability and cost-to-go supervision examples.

B Student Planning Representation Details

B.1 Student Architecture Details

Table 10 summarizes the architectural details of the proposed student planning representation pipeline, including the semantic distillation module, traversability prediction head, and goal-conditioned cost-to-go head.

The semantic distillation module refines the frozen LoftUp feature representations through lightweight teacher-guided semantic alignment during training. The resulting grounded representation F_{ground} is used only during training, while the student directly uses the projected representation F_{proj} during inference without requiring teacher supervision. Figure 9 visualizes the feature distributions of the raw LoftUp features, projected student features, and grounded representations using t-SNE. Compared with the raw LoftUp features, the projected and grounded representations exhibit more structured terrain-aware semantic grouping.

Beyond semantic grounding, and the traversability map prediction, the proposed planning representation also produces explicitly goal-conditioned cost-to-go predictions. Figure 10 shows representative examples where different goal locations over the same traversability map generate different dense cost-to-go representations, demonstrating that the planning representation dynamically adapts according to the target goal location and surrounding traversability structure.

Table 10: Architectural details of the student planning representation pipeline.

Module	Component	Configuration
Encoder	Visual backbone	Frozen LoftUp with DINOv2-S
	Input resolution	224×224
	Output feature dim.	384
Semantic Distillation	Projection block	Conv1×1 + GN + GELU
	Residual refinement	Conv3×3 + Conv1×1
	MLP refinement	Conv1×1 + GELU + Conv1×1
	Projection dim.	256
	Token pooling	Masked average pooling
	Token attention	4-head cross-attention
	Token injection	Additive masked injection
Traversability Head	Prediction head	1×1 Conv $\rightarrow$ ReLU $\rightarrow$ 1×1 Conv $\rightarrow$ ReLU $\rightarrow$ 1×1 Conv
	Hidden channels	$128 \rightarrow 64$
	Output activation	Sigmoid
Cost-to-Go Head	Cost encoder	3×3 Conv + BN + GELU + residual conv
	Coordinate encoder	3×3 Conv + BN + GELU
	Goal conditioning	FiLM modulation
	Context refinement	Lightweight cross-attention
	Cost fusion	Gated traversability fusion
	Residual refinement	ResidualConv $\times2$
	Prediction layer	1×1 Conv + Sigmoid

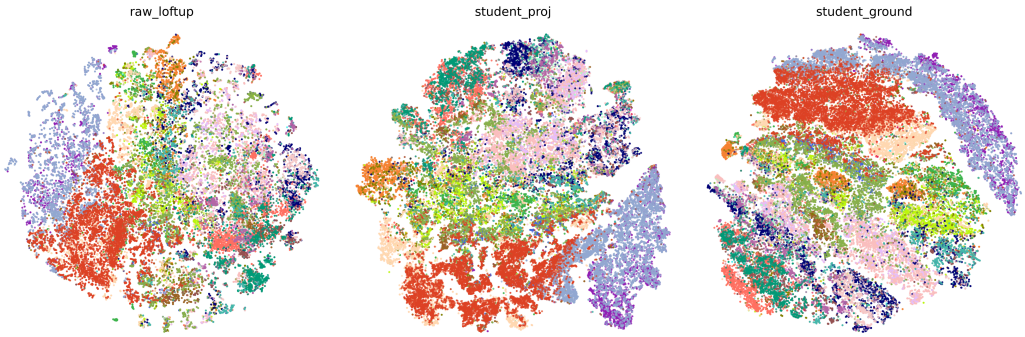

Figure 9: t-SNE visualization of the feature representations before and after semantic distillation.

B.2 Training Details

The proposed model is trained using PyTorch on a single NVIDIA H100 GPU. During training, the LoftUp encoder remains frozen, while the semantic distillation module, traversability prediction head, and goal-conditioned planning head are optimized jointly using AdamW. To ensure deterministic and reproducible validation, the teacher-generated traversability maps, sampled goals, and corresponding cost-to-go targets are precomputed and cached prior to validation. The model is trained for 40,000 iterations, after which additional training iterations provide only marginal performance improvements.

Table 11: Training hyperparameters.

Parameter	Value
Input resolution	224×224
Optimizer	AdamW
Learning rate	1×10^{-4}
Weight decay	1×10^{-4}
Batch size	2
Gradient clipping	1.0
Training iterations	40k
Loss weights (λ_1, λ_2)	$(0.5, 0.35)$

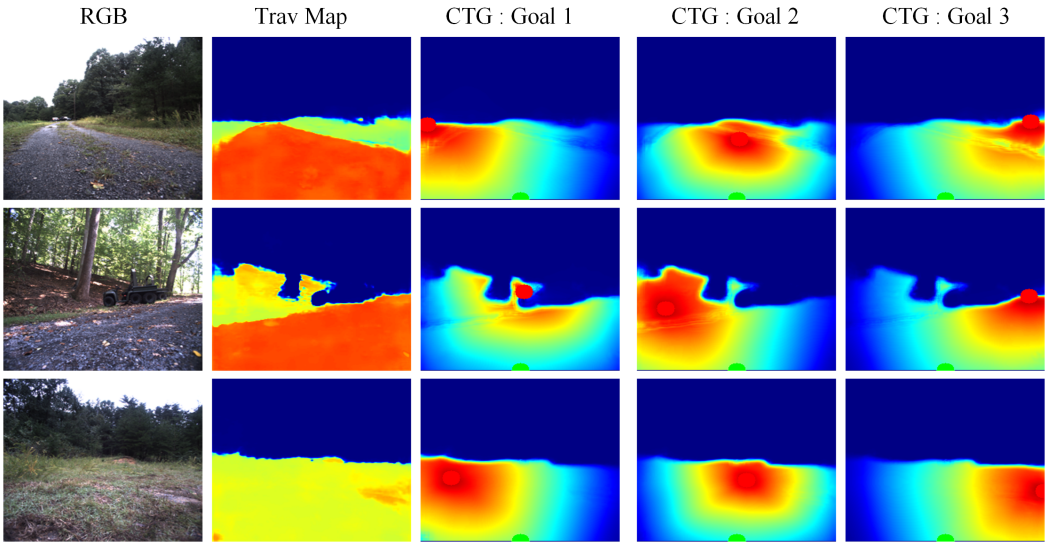

Figure 10: Goal-conditioned cost-to-go prediction examples. Different goal locations over the same traversability map produce different dense cost-to-go representations, demonstrating adaptive goal-aware planning behavior. Red circles denote goal locations, while green circles denote the start location.

C Representation Learning Ablations

We perform extensive ablation studies on the RUGD validation set to analyze the contribution of the proposed semantic distillation and goal-conditioned planning representation modules. Since the proposed framework jointly learns traversability-aware semantic representations and dense planning representations, the ablations primarily focus on the learned traversability representations and planning representations quality rather than downstream navigation performance.

We first analyze the contribution of the semantic distillation module by comparing the full model with a variant without semantic distillation. We then evaluate different backbone feature representations, including the utilized LoftUp features, DINO-only features, and a ResNet-based encoder. Next, we analyze the semantic distillation module through multiple token refinement and token injection variants, including removing cross-attention, using teacher-only or student-only refinement, and different token injection strategies. We additionally study the effect of simplifying the projection block through MLP and residual removal variants. Finally, we evaluate the contribution of the proposed goal-conditioned planning head through FiLM conditioning, cross-attention, and traversability fusion ablations.

For traversability prediction, we report mean absolute error (MAE), root mean squared error (RMSE), intersection-over-union at threshold 0.5 (IoU@0.5), and Kendall rank correlation. For the cost-to-go prediction, we additionally report RMSE, MAE, Pearson correlation between predicted and target cost-to-go maps, and gradient cosine similarity, which measures directional consistency between predicted and target planning gradients.

The semantic distillation study shows that removing semantic distillation consistently degrades both traversability and cost-to-go prediction quality, validating the effectiveness of the proposed teacher-guided semantic grounding process. The backbone study further demonstrates that the proposed LoftUp representation consistently outperforms both DINO-only and ResNet-based features across most traversability and planning metrics, indicating that high-resolution fine-grained feature representations are beneficial for dense traversability prediction and goal-conditioned planning.

The semantic distillation ablations demonstrate that both teacher-guided refinement and token reinjection contribute to improved representation learning for traversability-aware planning. In particular, the teacher-guided interaction consistently improves planning-oriented metrics such as Kendall correlation and cost-to-go correlation, indicating better global structural consistency in the learned

Table 12: Representation learning ablation studies for semantic distillation and goal-conditioned planning. Lower is better for ↓ metrics, while higher is better for ↑ metrics. Best and second-best results are highlighted within each ablation group.

	Traversability Metrics				Planning Metrics			
Model	MAE↓	RMSE↓	IoU@0.5↑	Kendall↑	RMSE↓	MAE↓	Corr↑	GradCos↑
Semantic Distillation								
W Distillation	**0.0374**	**0.0909**	**0.9336**	**0.9606**	**0.1008**	**0.0380**	**0.9523**	0.3653
W/O Distillation	0.0391	0.0959	0.9261	0.9580	0.1014	0.0387	0.9351	**0.3666**
Backbone Study								
LoftUp	**0.0374**	**0.0949**	**0.9336**	**0.9586**	**0.1008**	**0.0380**	**0.9372**	0.3653
DINO Only	0.0396	0.1010	0.9234	0.9543	0.1031	0.0386	0.9318	**0.3655**
ResNet	0.0448	0.1140	0.9125	0.9404	0.1118	0.0423	0.9239	0.3720
Semantic Distillation Ablations								
W/o Cross (Teacher Only)	0.0390	0.0948	0.9313	0.9594	0.1014	0.0394	0.9353	0.3662
w/o Cross (Student Only)	0.0381	0.0960	**0.9348**	0.9563	0.1011	0.0380	0.9360	**0.3695**
Injection (Replace)	**0.0368**	**0.0943**	0.9330	**0.9596**	**0.1006**	**0.0372**	**0.9363**	0.3628
No Injection	0.0387	0.0974	**0.9348**	0.9548	0.1012	0.0377	0.9343	0.3667
Injection (Gating)	0.0369	0.0946	0.9332	0.9592	0.1012	0.0377	0.9351	0.3650
Projection Block Ablations								
Proj (w/o MLP)	0.0375	**0.0941**	**0.9322**	**0.9598**	**0.1006**	0.0382	0.9356	0.3627
Proj (Identity Only)	0.0379	0.0943	0.9290	0.9595	0.1008	0.0379	**0.9377**	**0.3646**
Proj (Without Residual)	**0.0369**	0.0943	**0.9322**	**0.9598**	0.1008	**0.0374**	0.9355	0.3610
Goal-Conditioned Planning Head Ablations								
No FiLM	**0.0377**	**0.0959**	0.9334	**0.9581**	**0.1007**	0.0381	**0.9367**	**0.3635**
No Cross Attention	0.0387	0.0966	0.9341	0.9557	0.1009	0.0388	0.9361	0.3652
No Traversability Fusion	0.0382	0.0969	**0.9351**	0.9555	0.1008	**0.0374**	0.9360	0.3655

planning representation. Interestingly, the student-only and no-injection variants remain competitive in certain traversability metrics, suggesting that the semantic alignment objective itself already provides strong regularization for learning planning-aware representations even without explicit spatial token reinjection. Nevertheless, the full teacher-guided formulation achieves more balanced performance across both traversability estimation and goal-conditioned planning metrics, supporting the effectiveness of the proposed semantic refinement process.

Similarly, the projection block ablations demonstrate relatively stable performance across architectural simplifications, indicating that the semantic refinement process is not overly sensitive to the exact projection architecture. Even lightweight projection variants remain competitive, suggesting that the primary performance gains mainly originate from the semantic distillation objective and planning-aware representation refinement rather than from a specific projection design choice.

Finally, the goal-conditioned planning head ablations show that FiLM conditioning, cross-attention, and traversability fusion each contribute complementary improvements for dense cost-to-go prediction. Removing these components consistently reduces planning quality, supporting the effectiveness of the proposed goal-conditioned planning representation for adaptive goal-aware planning.

D Goal Projection and Navigation Towards the Goal

Given a global point-goal defined in the map frame, we continuously project the target location into the image space using the current robot pose together with a ground-plane camera projection model. Since the proposed framework performs local planning directly in image space, the projected target does not necessarily correspond to the final global destination itself. For distant targets outside or near the boundary of the current camera field of view, the projection instead produces a locally observable image-space subgoal. As the robot moves and its relative pose to the global target changes,

the projected subgoal progressively shifts through the image, resulting in a receding-horizon navigation behavior directly in image space (Figure 12 left).

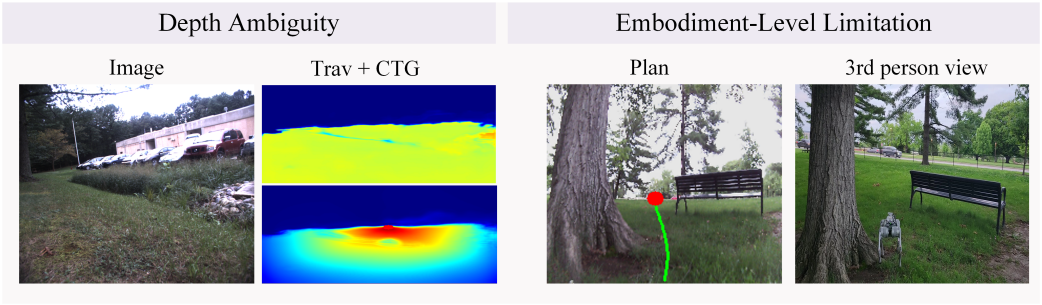

Figure 11: Representative limitations.

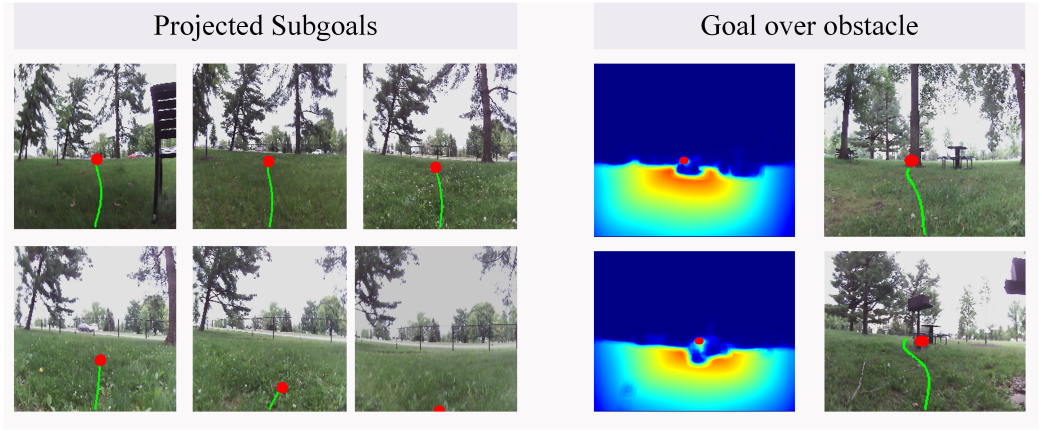

Figure 12: Projected image-space goal behavior during real-world deployment. Left: progressively shifting locally observable subgoals generated from distant global navigation targets. Right: projected goals may overlap with visually non-traversable regions due to perspective projection and limited scene visibility; the planner instead uses nearby feasible image-space regions for local navigation.

In practice, the projected goal may occasionally fall onto visually non-traversable regions or foreground obstacles due to perspective projection, limited scene visibility, and the simplified ground-plane assumption used during image-space goal construction. Since this projection does not explicitly model obstacle geometry or depth discontinuities, distant targets may project onto structures such as trees, poles, vegetation, or scene boundaries in the image plane. We therefore verify the projected target using the predicted traversability map before constructing the goal-conditioned planning representation. If the projected pixel is classified as non-traversable, the target is shifted to the nearest feasible pixel in image space. This corrected target is subsequently used for cost-to-go prediction and downstream planning, preventing the robot from being directly guided toward an image-space obstacle.

As the robot deviates around foreground obstacles, the relative direction toward the global target and its corresponding image-space projection are continuously updated according to the current robot pose and heading. Consequently, new locally observable subgoals are progressively generated throughout motion, allowing the planner to recover toward the final global destination after locally avoiding obstructed or cluttered regions. This behavior enables the proposed framework to perform closed-loop local navigation even when the direct image-space projection of a distant goal is temporarily infeasible.

E VALUE-GUIDED NAVIGATION DETAILS

E.1 GRAPH-BASED VALUE-GUIDED PLANNER

The graph-based planner operates directly over an 8-connected image-space grid. The robot start location s_0 is initialized at the bottom-center of the image, while the projected point goal $\bar{g}_t$ defines the target location. The predicted traversability map T_t determines the feasible search region, with pixels below the traversability threshold treated as non-traversable.

For two neighboring pixels u and v, the geometric transition cost is defined as

$$d(u,v) = \begin{cases} 1, & \text{for horizontal or vertical transitions,} \\ \sqrt{2}, & \text{for diagonal transitions.} \end{cases} \tag{2}$$

In addition to the geometric transition cost, the planner incorporates the predicted cost-to-go V_t, an obstacle-clearance cost C_t^{clr}, and a temporal path-consistency cost C_t^{temp}. The resulting transition cost is

$$c_t(u,v) = d(u,v) + \lambda_V V_t(v) + \lambda_C C_t^{\mathrm{clr}}(v) + \lambda_P C_t^{\mathrm{temp}}(v), \tag{3}$$

where λ_V, λ_C, and λ_P control the relative contributions of the corresponding terms. The clearance cost is obtained from the distance to the nearest non-traversable region, assigning larger costs to pixels closer to obstacles. The temporal consistency term is constructed from the previous planned path and penalizes large deviations from the preceding image-space route.

The accumulated route cost for reaching a neighboring pixel v from u is updated as

$$G(v) = G(u) + c_t(u,v). \tag{4}$$

For goal-directed search, nodes are prioritized according to

$$F(v) = G(v) + \lambda_H \left\| v - \bar{g}_t \right\|_2, \tag{5}$$

where the second term provides Euclidean goal guidance. Setting $\lambda_H = 0$ removes this guidance and yields the heuristic-free graph-search variant used in our planner analysis.

Algorithm 1 summarizes the complete graph-based planning procedure.

If the exact projected goal cannot be reached because of blocked or disconnected image regions, the planner selects the reachable endpoint with the smallest image-space distance to the requested goal, using the predicted cost-to-go to resolve equally distant candidates. The reconstructed waypoint sequence is subsequently smoothed before being passed to the shared closed-loop controller.

E.2 GRADIENT-BASED ROLLOUT PLANNER

The gradient-based rollout planner directly follows the local descent direction of the learned planning representation without constructing an explicit search graph or sampling multiple candidate trajectories. As in the graph-based formulation, the predicted traversability map T_t defines hard obstacles and is additionally used to construct the obstacle-clearance cost C_t^{clr}.

Prior to gradient computation, the learned cost-to-go representation is lightly smoothed to reduce local prediction noise. We denote the resulting cost-to-go field as $\widetilde{V}_t$. The continuous planning potential is then defined as

$$\Phi_t^{\mathrm{grad}}(x) = \lambda_V \widetilde{V}_t(x) + \lambda_C C_t^{\mathrm{clr}}(x), \tag{6}$$

where λ_V and λ_C control the contributions of the learned cost-to-go and obstacle-clearance terms, respectively.

Starting from the bottom-center image location $p_0 = s_0$, the planner computes the normalized negative gradient of the planning potential,

$$d_k = -\frac{\nabla \Phi_t^{\mathrm{grad}}(p_k)}{\left\| \nabla \Phi_t^{\mathrm{grad}}(p_k) \right\|_2}, \tag{7}$$

Algorithm 1 Graph-Based Cost-to-Go-Guided Image-Space Planning

Input: $T_t, V_t, s_0, \bar{g}_t, \mathcal{P}_{t-1}$
Construct the feasible 8-connected image-space graph from T_t
Compute C_t^{clr} and C_t^{temp}
Initialize $G(x) \leftarrow \infty$ for all feasible pixels x
$G(s_0) \leftarrow 0$
$F(s_0) \leftarrow \lambda_H \|s_0 - \bar{g}_t\|_2$
Initialize the graph-search queue with s_0
while queue is not empty **do**
    Pop u with minimum $F(u)$
    if $u = \bar{g}_t$ **then**
        break
    end if
    for each feasible 8-connected neighbor v of u **do**
        $\widetilde{G}(v) \leftarrow G(u) + c_t(u, v)$
        if $\widetilde{G}(v) < G(v)$ **then**
            $G(v) \leftarrow \widetilde{G}(v)$
            $\pi(v) \leftarrow u$
            $F(v) \leftarrow G(v) + \lambda_H \|v - \bar{g}_t\|_2$
            Insert or update v in the queue
        end if
    end for
end while
Reconstruct the path using predecessor map π
Apply local path smoothing
return $\mathcal{P}_t$

which provides the local direction of decreasing predicted navigation cost. To reduce abrupt directional changes between consecutive rollout steps, a small momentum term is applied:

$$\hat{d}_k = \mathrm{norm}\left((1-\mu)d_k + \mu\hat{d}_{k-1}\right), \tag{8}$$

where μ denotes the momentum coefficient and $\mathrm{norm}(\cdot)$ denotes unit normalization. The next rollout point is then obtained using a fixed image-space step size η:

$$p_{k+1} = p_k + \eta\hat{d}_k. \tag{9}$$

A rollout step is accepted only if the resulting point remains inside the image and does not enter a hard obstacle defined by T_t. The rollout terminates when the projected goal is reached, the gradient becomes degenerate, no meaningful reduction in planning cost is observed, the trajectory repeatedly revisits the same region, or the maximum rollout length is reached. If the exact goal is not reached, the planner retains the useful prefix ending at the lowest-cost point encountered during the rollout. The resulting waypoint sequence is subsequently smoothed before being passed to the shared closed-loop controller.

Algorithm 2 summarizes the procedure.

E.3 Gradient-Guided Sampling Planner

The gradient-guided sampling planner combines the local directional information contained in the learned cost-to-go representation with short-horizon trajectory sampling. Unlike the graph-based planner, it does not construct an explicit search graph. Unlike direct gradient rollout, it does not commit to a single local descent direction. Instead, multiple candidate rollouts are evaluated around the local cost-to-go gradient at each planning step.

Algorithm 2 Gradient-Based Cost-to-Go Rollout

Input: $T_t, V_t, s_0, \bar{g}_t$
Construct hard-obstacle mask from T_t
Compute obstacle-clearance cost C_t^{clr}
Smooth V_t to obtain $\widetilde{V}_t$
$\Phi_t^{\mathrm{grad}} \leftarrow \lambda_V \widetilde{V}_t + \lambda_C C_t^{\mathrm{clr}}$
Compute $\nabla \Phi_t^{\mathrm{grad}}$
$p_0 \leftarrow s_0$
Initialize path $\mathcal{P}_t \leftarrow \{p_0\}$
for $k = 0, \ldots, K-1$ **do**
    if $\|p_k - \bar{g}_t\|_2 \le \epsilon_g$ **then**
        Append $\bar{g}_t$ and **break**
    end if
    $d_k \leftarrow -\nabla \Phi_t^{\mathrm{grad}}(p_k) / \|\nabla \Phi_t^{\mathrm{grad}}(p_k)\|_2$
    if gradient magnitude is too small **then**
        break
    end if
    $\hat{d}_k \leftarrow \mathrm{norm}((1-\mu) d_k + \mu \hat{d}_{k-1})$
    $p_{k+1} \leftarrow p_k + \eta \hat{d}_k$
    if p_{k+1} is outside the image or lies on an obstacle **then**
        break
    end if
    Append p_{k+1} to $\mathcal{P}_t$
    if no sufficient cost decrease or repeated revisiting is detected **then**
        break
    end if
end for
Retain the lowest-cost valid rollout prefix if the goal is not reached
Smooth the resulting path
return $\mathcal{P}_t$

The learned cost-to-go map is first lightly smoothed to stabilize its spatial gradient. At the current image-space location p_k, the nominal descent direction is computed as

$$d_k = -\frac{\nabla \widetilde{V}_t(p_k)}{\left\| \nabla \widetilde{V}_t(p_k) \right\|_2}, \tag{10}$$

where $\widetilde{V}_t$ denotes the smoothed cost-to-go representation. A set of K angular offsets $\{\theta_i\}_{i=1}^K$ is then uniformly sampled within a symmetric angular range around d_k.

Each angular offset defines a short candidate rollout. Starting from $q_0^i = p_k$, the local CTG descent direction is recomputed at every preview step and rotated by θ_i:

$$q_{h+1}^i = q_h^i + \eta_s R(\theta_i) \left(-\frac{\nabla \widetilde{V}_t(q_h^i)}{\left\| \nabla \widetilde{V}_t(q_h^i) \right\|_2} \right), \tag{11}$$

where $R(\theta_i)$ denotes a 2-D rotation and η_s is the preview step size. Candidate rollouts that leave the image or intersect a hard obstacle defined by T_t are discarded.

For each valid candidate, the rollout score is computed as

$$J_i = \lambda_V \frac{1}{H} \sum_{h=1}^{H} \widetilde{V}_t(q_h^i) + \lambda_C \frac{1}{H} \sum_{h=1}^{H} C_t^{\mathrm{clr}}(q_h^i), \tag{12}$$

where H is the sampling horizon. We use $\lambda_V = 3$ and $\lambda_C = 4$. Since all candidate rollouts use the same number of fixed-length preview steps, the geometric transition cost is constant across candidates and therefore does not affect their ranking.

The candidate with the lowest score is selected, but only its first direction is executed:

$$p_{k+1} = p_k + \eta\, d_k^*, \qquad i^* = \arg\min_i J_i, \tag{13}$$

where d_k^* is the first-step direction of the winning rollout and η is the executed step size. The sampling process is then repeated from p_{k+1}, yielding a receding-horizon image-space path toward the projected goal.

Algorithm 3 summarizes the procedure.

Algorithm 3 Gradient-Guided Cost-to-Go Sampling

Input: $T_t, V_t, s_0, \bar{g}_t$
Construct hard-obstacle mask from T_t
Compute clearance cost C_t^{clr}
Smooth V_t to obtain $\widetilde{V}_t$
Compute $\nabla \widetilde{V}_t$
$p_0 \leftarrow s_0$
Initialize path $\mathcal{P}_t \leftarrow \{p_0\}$
for $k = 0, \ldots, K_{\max} - 1$ **do**
  if $\|p_k - \bar{g}_t\|_2 \le \epsilon_g$ **then**
    Append $\bar{g}_t$ and **break**
  end if
  Compute local descent direction $d_k \leftarrow -\nabla \widetilde{V}_t(p_k) / \|\nabla \widetilde{V}_t(p_k)\|_2$
  Generate angular offsets $\{\theta_i\}_{i=1}^K$ around d_k
  for each offset θ_i **do**
    Initialize candidate $q_0^i \leftarrow p_k$
    for $h = 0, \ldots, H - 1$ **do**
      Recompute local CTG descent direction at q_h^i
      Rotate direction by θ_i
      Roll out one preview step to q_{h+1}^i
      if candidate leaves image or intersects an obstacle **then**
        Mark candidate invalid and **break**
      end if
    end for
    if candidate is valid **then**
      Compute $J_i = \lambda_V \overline{V}_i + \lambda_C \overline{C}_i^{\mathrm{clr}}$
    end if
  end for
  $i^* \leftarrow \arg\min_i J_i$
  if no valid candidate exists **then**
    break
  end if
  Execute only the first step of candidate i^*
  Append the new waypoint to $\mathcal{P}_t$
  if stalling or repeated revisiting is detected **then**
    break
  end if
end for
Retain the valid rollout prefix if the goal is not reached
Smooth the resulting path
return $\mathcal{P}_t$

Given a planned path of N image-space points, the look-ahead index is selected as

$$k_{\mathrm{look}} = \min\left(\max\left(10, \left\lfloor \frac{N}{4} \right\rfloor\right), N - 1\right), \tag{14}$$

and the corresponding image-space waypoint is passed to the downstream controller. This intermediate path target preserves the local structure of the planned route while providing smoother closed-loop tracking than directly steering toward the terminal image-space goal.

E.4 IMAGE-SPACE TRACKING CONTROLLER

The selected look-ahead waypoint is converted into linear and angular velocity commands using a lightweight image-based controller. Let $q_t = (x_t, y_t)$ denote the target pixel and W denote the image width. During nominal navigation, the horizontal image-space tracking error relative to the image center is computed as

$$e_x = \frac{x_t - W/2}{W/2}. \tag{15}$$

A small steering deadband is applied around the image center to suppress oscillatory heading corrections caused by minor frame-to-frame variations in the predicted target. The commanded angular velocity is computed using proportional image-space steering:

$$\omega_t = \begin{cases} 0, & |e_x| < \delta_e, \\ \mathrm{clip}\left(-k_\omega e_x, -\omega_{\max}, \omega_{\max}\right), & \text{otherwise}, \end{cases} \tag{16}$$

where k_ω denotes the steering gain and δ_e is the image-space deadband.

The forward velocity is adaptively reduced as the steering magnitude increases according to

$$v_t = \mathrm{clip}\left(v_{\max}\left[1 - \min\left(\alpha_v |e_x|, \rho_v\right)\right], 0, v_{\max}\right), \tag{17}$$

where $v_{\max}$ denotes the maximum nominal forward velocity, α_v controls the steering-dependent velocity reduction, and ρ_v limits the maximum reduction. This formulation allows the robot to maintain higher forward velocity when the selected waypoint lies near the image center while slowing down during larger steering corrections.

The global robot pose is not used for nominal image-space path tracking and is instead used for goal-distance monitoring and near-goal stabilization. When the robot enters a $1.0\,\mathrm{m}$ neighborhood of the global target, the controller switches from image-space tracking to direct goal-heading stabilization. Given the robot position (x_r, y_r), yaw θ_r, and global target (x_g, y_g), the desired heading is computed as

$$\theta_g = \mathrm{atan2}\left(y_g - y_r, x_g - x_r\right), \tag{18}$$

with the wrapped heading error

$$e_\theta = \mathrm{wrap}\left(\theta_g - \theta_r\right). \tag{19}$$

The near-goal angular and linear velocity commands are then computed as

$$\omega_t = \mathrm{clip}\left(k_\theta e_\theta, -1, 1\right), \tag{20}$$

and

$$v_t = \mathrm{clip}\left(k_d d_g, 0, v_{\mathrm{near}}\right), \tag{21}$$

where d_g denotes the Euclidean distance to the global target.

Once the robot reaches the final stopping threshold, a short low-speed reverse command is applied before repeatedly publishing zero-velocity commands to improve stopping stability. The controller additionally monitors the freshness of the image-space target and stops the robot whenever a valid target has not been received within the predefined timeout. Table 13 summarizes the primary planning and control parameters used during deployment.

F METRIC DEFINITION FOR EVALUATING NAVIGATION

F.1 OPEN-LOOP PLANNING METRIC DEFINITIONS

We evaluate open-loop planning quality in image space by comparing the predicted path $s_{1:N}$ generated from the student-predicted dense cost-to-go representation against the reference path $s^*_{1:M}$ generated from the teacher-generated reference cost-to-go representation under the identical planning setup. Specifically, both paths are obtained using the same value-guided planner to isolate the effect of the learned planning representations.

Table 13: Planner and controller parameters used during deployment.

Parameter	Value
Planner search resolution (Simulation)	224×224
Planner search resolution (Real)	64×64
Planner connectivity	8-connected
Planner traversability threshold	0.40
Projected-goal feasibility threshold	0.45
Cost-to-go weight λ_V	3.0
Clearance weight λ_C	4.0
Temporal consistency weight λ_P	0.2
Clearance radius	25 pixels
Temporal corridor radius	30 pixels
Path smoothing iterations	50
Maximum linear velocity $v_{\max}$	0.6 m/s
Angular steering gain k_ω	0.8
Angular velocity range	$[-1.0, 1.0]$ rad/s
Steering deadband δ_e	0.05
Velocity scaling factor α_v	1.5
Maximum velocity reduction ρ_v	0.9
Near-goal threshold	1.0 m
Near-goal angular gain k_θ	1.2
Near-goal distance gain k_d	0.4
Maximum near-goal velocity v_{near}	0.3 m/s
Final stopping threshold	0.5 m
Target timeout	0.4 s
Control frequency	20 Hz

Path Efficiency: Path efficiency measures how closely the predicted path length matches the teacher reference path length under successful goal reaching:

$$\mathrm{PE} = \mathbb{I}(e_{\text{end}} < \delta)\frac{L(s^*_{1:M})}{\max(L(s^*_{1:M}), L(s_{1:N}))},$$

where $L(\cdot)$ denotes path length, e_{end} is the endpoint error, and δ is the success threshold. In our evaluation, a predicted path is considered successful if the final endpoint error is below $\delta = 10$ pixels. Higher path efficiency values indicate paths that are both successful and closer in length to the teacher reference path.

Endpoint Error: Endpoint error measures the Euclidean distance between the final point of the predicted path and the final point of the teacher reference path:

$$e_{\text{end}} = \|s_N - s^*_M\|_2 .$$

Path Deviation: Path deviation measures the average point-wise distance between the predicted and reference paths after truncating both paths to the same minimum length:

$$D_{\text{path}} = \frac{1}{K}\sum_{i=1}^{K} \|s_i - s^*_i\|_2 , \quad K = \min(N, M).$$

Path Smoothness: Path smoothness is computed as the mean angular change between consecutive path segments:

$$S = \frac{1}{N-2}\sum_{i=2}^{N-1} \cos^{-1}\left(\frac{(s_i - s_{i-1})\cdot(s_{i+1} - s_i)}{\|s_i - s_{i-1}\|_2\|s_{i+1} - s_i\|_2}\right).$$

Lower smoothness values indicate smoother paths with smaller directional changes.

F.2 Closed-loop Navigation Metric Definitions

For closed-loop navigation evaluation in simulation, we report Success Rate, Success weighted by normalized inverse Path Length (SPL), and Risk Score.

Success Rate: An episode is considered successful if the robot reaches the target goal within a Euclidean distance threshold of (0.5 m) without collision. The success rate is computed as $(\mathrm{SR} = \frac{1}{M}\sum_{i=1}^{M} S_i), where(S_i \in 0, 1)$ indicates whether episode (i) is successful and (M) denotes the total number of evaluation episodes.

Success weighted by normalized inverse Path Length (SPL): To evaluate path efficiency during successful closed-loop navigation, we compute SPL only over successful episodes: $(\mathrm{SPL} = \frac{1}{N_s}\sum_{i=1}^{N_s} \frac{\ell_i}{\max(p_i,\ell_i)})$, where (N_s) denotes the number of successful episodes, (ℓ_i) is the shortest start-to-goal distance for episode (i), and (p_i) is the executed trajectory length. Higher values indicate trajectories closer to the shortest feasible path.

G NAVIGATION DEMONSTRATION

We additionally provide several real-world deployment examples across different indoor and outdoor navigation scenarios. The demonstrated environments include narrow trails, mixed pavement–grass terrain, park-like scenes with trees and benches, paved pathways, cluttered outdoor areas, and indoor corridors. Across these deployments, the monocular RGB observation is used to predict dense traversability and goal-conditioned cost-to-go representations for local image-space planning, while global goal projection and goal monitoring rely on an external localization system. The examples further demonstrate the practical deployment of VGN across diverse scene layouts, terrain types, obstacle configurations, and illumination conditions.

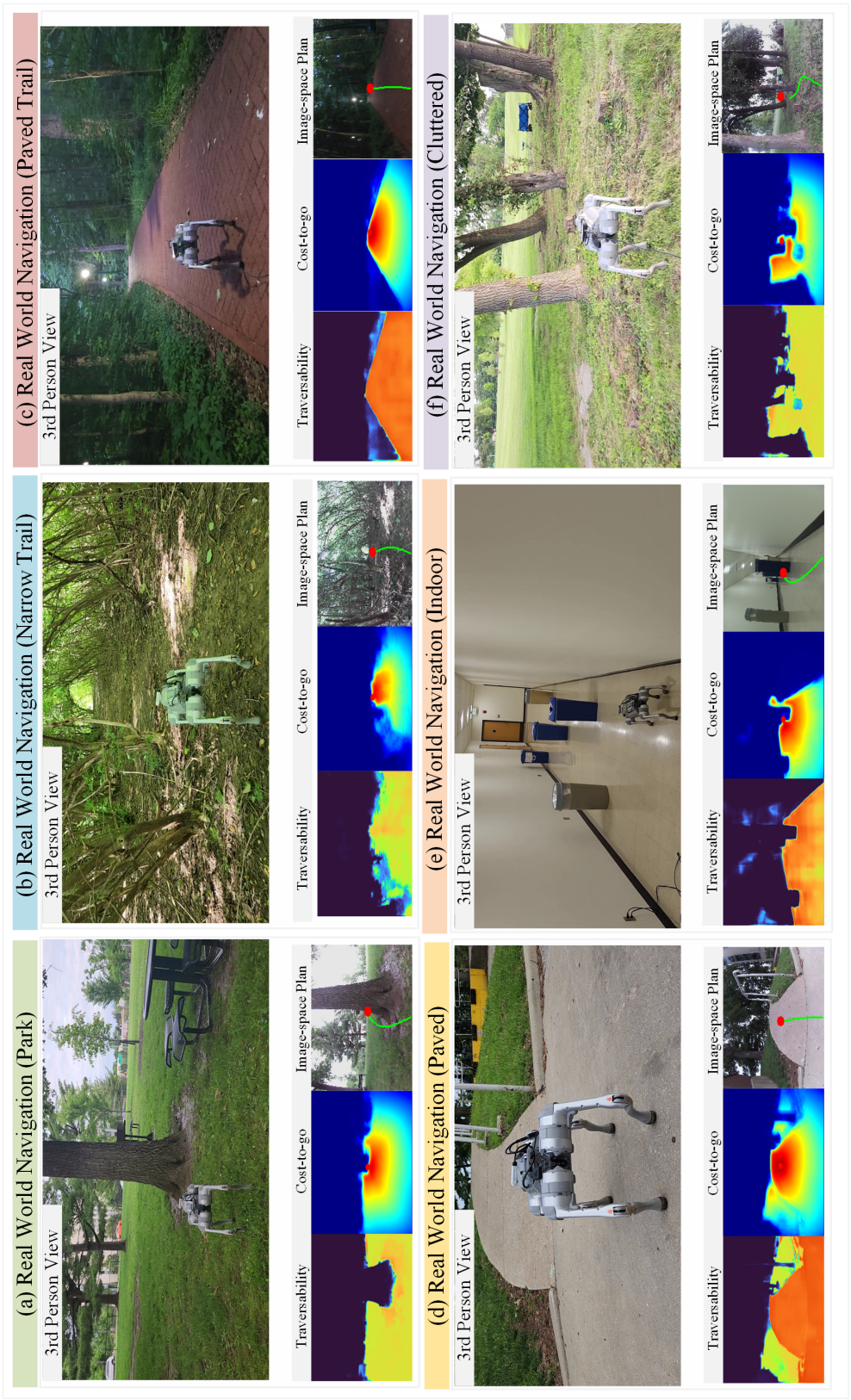

Figure 13: Real-world closed-loop deployment.